Immersions - Musik in Künstlichen Ohren

Vincent Herrmann
vincent.herrmann@web.de
Institut für Musikwissenschaft und Musikinformatik, Hochschule für Musik Karlsruhe
Karlsruhe, Deutschland

ZUSAMMENFASSUNG

Immersions ist eine Performance Elektronischer Musik, die in Echtzeit die inneren Vorgänge eines neuronalen Netzes sonifiziert. Dabei werden durch ein Optimierungsverfahren Klänge generiert, die bestimmte Bereiche des Netzwerks aktivieren. So wird hörbar, wie Musik in diesem künstlichen Ohr klingt. Zusätzlich werden die Vorgänge auch visualisiert, um eine möglichst immersive Erfahrung zu schaffe, die es erlaubt einzutauchen in die Tiefe Künstlicher Intelligenz.

KEYWORDS

neural networks, music interfaces, sound generation

1 EINFÜHRUNG

Seit nun etwa fünf Jahren hält künstliche Intelligenz, insbesondere in Form von neuronalen Netzen und sogenanntem Deep Learning, Einzug in unseren Alltag. Zuerst waren die Erfolge hauptsächlich auf visuelle Probleme beschränkt, zunehmend weiten sie sich aber auch auf andere Bereiche, wie zum Beispiel die Klangverarbeitung aus: Google generiert so die Stimme des Sprachassistenten, Spotify wertet Millionen Songs aus um sie zu automatisch personalisierten Playlists zusammenzustellen, inzwischen komponieren künstliche neuronale Netze sogar Popsongs und Filmmusik.

Oft ist es selbst für Experten schwierig zu erkennen, was genau in diesen komplexen selbstlernenden Systemen vor sich geht und wie sie zu diesen bemerkenswerten Ergebnissen kommen. Für bildverarbeitende Netze existieren Techniken um die internen Repräsentationen der Netze anschaulich zu machen - das von Google entwickelte DeepDream-Verfahren [6] generiert so bizarre psychedelische, auf uns aber auch oft erstaunlich vertraut wirkende Bilder.

Mit ähnlichen Verfahren versuchen wir hier das Innere von klangverarbeitenden Netze hörbar zu machen. Aus verschiedenen technischen Gründen ist die Umsetzung für Musik recht kompliziert, aber gerade auch aus künstlerischer Sicht hochinteressant.

Die Grundidee ist im Wesentlichen folgende: Man geht von einem beliebigen Klangschnipsel aus (z.B. einem Drum-Loop). Dieser wird durch ein Optimierungsverfahren so verändert, dass er einen bestimmten Bereich im neuronalen Netz besonders aktiviert. Der veränderte Clip kann dann als Basis für eine neue Optimierung mit einem anderen Bereich im Netz als Kriterium verwendet werden. So erhält man Klänge, die das neuronale Netz frei assoziativ generiert.

Eine der wichtigsten Eigenschaften von neuronalen Netzen ist, dass sie Informationen in verschiedenen Abstraktionsgraden verarbeiten. Entsprechend kann eine Aktivierung in einem bestimmten Bereich des Netzes zum Beispiel mit einfachen kurzen Tönen oder Geräuschen einhergehen, in einem anderen Bereich mit komplexeren musikalischen Phänomenen wie Phrase, Rhythmus oder Tonart. Worauf genau ein neuronales Netz "hört", hängt sowohl von den Daten ab, anhand derer es gelernt hat, als auch von der Aufgabe, für die es spezialisiert ist. Das heißt, für unterschiedliche Netze klingt Musik auch unterschiedlich.

All das wird mit diesem Projekt direkt erfahrbar gemacht. Dafür entwickeln wir ein Setup, mit dem - einem DJ-Mischpult nicht unähnlich - die Generierung und Steuerung der Klänge in Echtzeit möglich ist. Als Material für die Live-Performance dienen dann nicht Songs oder vorprogrammierte Loops, sondern Halluzinationen neuronaler Netze.

Im Folgenden präsentieren wir die verschiedenen Teilaspekte des Projekts: Constrastive Predictive Coding ist ein Verfahren um neuronale Netze unüberwacht zu trainieren, das angelehnt ist an die Verarbeitung von Reizen im Gehirn. Im Abschnitt Netzwerk Architektur wird das konkrete neuronale Netz beschrieben, das den Ausgangspunkt der Performance darstellt. Unter dem Punkt Visualisierung erläutern wir, wie sich die Vorgänge im Netz intuitiv verständlich animieren lassen. Der Abschnitt Input Optimierung erklärt ein Verfahren, das es erlaubt mittels gradientenbasierter Optimierung Klänge zu erzeugen, welche das neuronale Netz auf bestimmte Weise anregen. Im letzten Teil, Live Performance, beschreiben wir schließlich, wie sich das Setup für die Performance konkret zusammensetzt.

MuC'19 Workshops, Hamburg, Deutschland

https://doi.org/10.18420/muc2019-ws-580

2 CONTRASTIVE PREDICTIVE CODING

Ausgangspunkt für dieses Projekt ist ein "künstliches Ohr", also ein neuronales Netz, das Klang verarbeitet. Es konnte gezeigt werden, dass Analogien bestehen zwischen Convolutional Neural Networks (ConvNets) und dem menschlichen auditiven Kortex [5]. In einem konkreten Experiment wird ein herkömmliches ConvNet für zwei unterschiedliche Klassifikationsaufgaben optimiert: das Erkennen von Wörtern und das Klassifizieren musikalischer Genres, beides anhand zweisekündiger Soundclips. Das resultierende Netzwerk spiegelt, durch ein lineares Mapping, die kortikalen Aktivierungen menschlicher Probanden, während sie dieselbe Aufgabe lösen, genauer wider als andere simplere Modelle.

Das Hörsystem des Menschen ist aber natürlich kein Klassifikationsnetzwerk, wir lernen nicht anhand expliziter Labels. Auch wenn es aus neuro- und kognitionswissenschaftlicher Sicht viele Standpunkte und offene Fragen gibt, ist doch ein aussichtsreicher Kandidat für eine Theorie des unüberwachten Lernens im Gehirn, insbesondere für Sinneswahrnehmungen, sogenanntes Predictive Coding [2]. Beim Predictive Coding werden zukünftige Reize vorhergesagt und dann mit den tatsächlich eingetroffenen Reizen verglichen. Das Lernen besteht darin, die Diskrepanz zwischen Vorhersage und Realität zu verringern. So muss sich ein Verständnis für das verarbeitende Signal entwickeln: Nur mit einem aussagekräftigen internen Modell davon, wie das Signal entstanden ist, kann zuverlässig vorhergesagt werden, wie es sich fortsetzen wird.

Eine Methode, um Predictive Coding zum unüberwachten Lernen für künstliche neuronale Netze anzuwenden, wird in [8] beschrieben. Beim Contrastive Predictive Coding benötigen wir zwei Netze: Ein Encoder-Netz, das ein komprimiertes Encoding des Signals erstellt, und ein autoregressives Modell, das mehrere aufeinanderfolgende Encodings zusammenfasst. Anhand des Outputs des autoregressiven Modells werden durch eine lineare Transformation zukünftige Encodings vorhergesagt. Praktisch funktioniert das, indem aus einer Auswahl von Möglichkeiten jedem Signal mit möglichst großer Konfidenz das richtige zukünftige Encoding zugeordnet werden muss. Dadurch wird die Transinformation (engl. Mutual Information) zwischen dem Output des autoregressiven Modells und den Encodings maximiert.

Die Tatsache, dass wir Contrastive Predictive Coding verwenden, erlaubt es uns die Daten frei auszusuchen, mit denen wir das Netz trainieren. Es sind keine aufwendig gelabelten Datasets nötig. Das wichtigste Modell für dieses Projekt wurde mit einem Dataset bestehend aus ca. 40 Stunden House-Mixes trainiert.

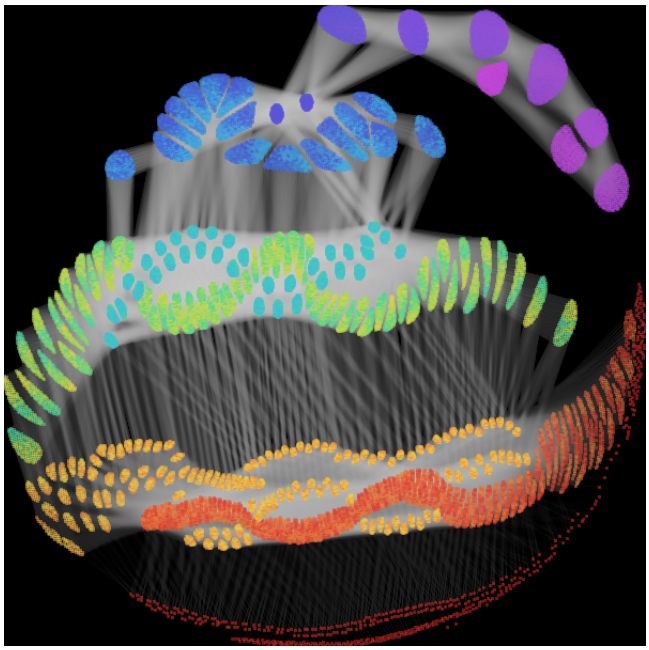

Abbildung 1: Visualisierung der Architektur von Encoder und Autoregressiven Modell zu einem bestimmten Zeitpunkt. Die roten Punkte am unteren Bildrand stellen den Input in Form eines Skalogramms dar. Der Großteil des Bildes wird vom Encoder eingenommen. Das autoregressive Modell ist in Lila- und Pinktönen rechts oben zu sehen.

3 NETZWERK ARCHITEKTUR

Als Input erhält das Encoder Netzwerk das Skalogramm eines gut viersekündigen Audioclips. Ein Skalogramm ist das Ergebnis einer Constant-Q- bzw. Wavelet-Transformation des Audiosignals und kommt der Repräsentation recht nahe, die von der Hörschnecke ans Gehirn weitergeleitet wird (vgl. [5]). Da es sich hierbei um eine 2D-Repräsentation des Signals handelt (Zeit und Tonhöhe), können wir ein herkömmliches 2D-ConvNet verwenden, wie es auch in der Bildverarbeitung üblich ist. Es ist allerdings nicht vollständig geklärt, ob Architekturen, die für Bilder optimiert wurden, auch am besten für andere 2D-Repräsentationen funktionieren. Anders formuliert - es ist unklar, wie effektiv der Induktive Bias klassischer ConvNets auch für Skalogramme ist. Wir verwenden als Encoder ein ResNet [3] mit ReLU-Nichtlinearitäten, Max-Pooling und Batch-Normalisierung. Zusätzlich zu den herkömmlichen Convolutions mit quadratischem Kernel verwenden wir auch auch Convolutions mit Kernels, die sich nur über die Tonhöhendimension erstrecken, und so Obertonspektren und Harmonien erkennen sollen.

Das Autoregressive Modell ist ebenfalls ein ResNet, allerdings mit 1D-Convolutions. Der Skalogramm Input ist ein Vektor bestehend aus 256 Tonhöhen mit einer zeitlichen

Auflösung von 125 Hz. D as Encoding hat 512 Kanäle und eine zeitliche Auflösung von 16 Hz. Das autoregressive Modell komprimiert den gesamten Input auf einen einzelnen 256-dimensionalen Vektor.

4 VISUALISIERUNG

Um zu verdeutlichen, was in dem Modell zu jedem Zeitpunkt vor sich geht, werden die Aktivierungen der künstlichen Neuronen visualisiert. Dabei bleibt die Zeitdimension in Echtzeit erhalten, was zu animierten Bildern führt. Die Neuronen müssen anhand ihrer Position im Netz (gegeben durch Schicht, Kanal und Tonhöhe) in einem 2D-Layout angeordnet werden. Die Neuronen sind, abhängig von dem Typus der Schicht (z.B. Convolutional oder Fully Connected), auf unterschiedliche Weise miteinander verbunden. Dadurch ergibt sich ein Graph-Layout Problem. Für das Finden eines Layouts für einen Graphen von solcher Größe (über 20000 Knoten und 20000000 Verbindungen) eignet sich die Multilevel-Force-Layout Methode [4]. Hierbei wird der Graph als physikalisches System modelliert, bei dem sich die Knoten wie durch elektrostatische Kraft gegenseitig abstoßen, verbundene Knoten sich aber wie durch Gummibänder anziehen. Um lokale Energieminima zu vermeiden, beginnt man mit wenigen Knoten und fügt schrittweise immer mehr Knoten hinzu, bis das Layout für den gesamten Graph berechnet ist. Abbildung 1 zeigt ein resultierendes Layout des hier verwendeten Netzes.

Ist das Layout einmal berechnet, kann der aktuelle Zustands des Netzes einfach dargestellt werden indem stark aktivierte Neuronen aufleuchten, andere aber dunkel bleiben.

5 INPUT OPTIMIERUNG

Heutige neuronale Netze sind komplett differenzierbar. Das heißt, der Einfluss jeder in den Berechnungen verwendeten Variablen auf darauffolgende Ergebnisse kann durch in Form des Gradienten festgestellt werden. Dadurch kann man diese so anpassen, dass sie ein Ergebnis minimieren oder maximieren. Beim Trainieren werden die Parameter des Netzes auf diese Weise schrittweise verändert, sodass der Fehler des Outputs minimiert wird.

Hat man ein trainiertes Netz, kann dieses Verfahren genutzt werden, um dieses Netz genauer zu untersuchen. Die verdeckten Aktivierungen sind natürlich abhängig vom Input. Daher können wir durch gradientenbasierte Optimierung Inputs generieren, die Aktivierungen bestimmter Neuronen maximieren [7]. Diese Inputs sind dann auch für uns Menschen direkt erfahrbar und zeigen, auf welche Reize des ausgewählten künstlichen Neuronen besonders ansprechen.

Neuronale Netze, zumindest wenn sie kein adversarial Training erfahren haben [1], sind anfällig für kleine Veränderungen im Input. Dort gibt es also lokale Optima die für den Menschen gar nicht wahrnehmbar sind und trotzdem die gewünschte Eigenschaft haben. Um dem vorzubeugen wenden wir mehrere Arten von Regularisierung an: zeitliche Verschiebung des Inputs, kleine Verschiebungen im Tonhöhenbereich, Maskierungen im Skalogramm, verrauschte Gradienten und De-Noising des Inputs. All diese Methoden machen das Input-Optimierungsverfahren in gewisser Weise schwieriger und erzwingen dadurch robustere, deutlichere und klarere Ergebnisse.

6 LIVE PERFORMANCE

Die generierten Audioclips haben eine Länge von ca. vier Sekunden. Damit eignen sie sich für eine Loop-basierte Live-Performance. Ein Clip entspricht dann z.B. zwei 4/4-Takten bei 120 BPM. Mit dem beschriebenen Optimierungsverfahren werden konstant neue Audioclips generiert. Wurde ein Clip zu Ende abgespielt, startet automatisch der aktuellste neu berechnete Clip. So ergibt sich ein akustisches Morphing, das sich aber auch durch die Visualisierung nachvollziehen lässt (jedes Aktivierungsmuster ergibt eigene Klänge). Als Ausgangspunkt für die Optimierung kann Stille oder Rauschen dienen, aber auch ein vorgefertigter Clip, der beispielsweise einen konkreten Rhythmus vorgibt.

Alle Aspekte des Verfahrens können in Echtzeit angepasst werden. Für die intuitive Kontrolle wurde ein GUI entwickelt und die wichtigsten Parameter durch einen MIDI-Controller steuerbar gemacht. Am wichtigsten ist die Auswahl der Aktivierungen, die maximiert werden sollen. Dafür verwenden wir einen Drehregler und sechs Fader. Mit dem Drehregler wählt man eine Schicht aus. Mit den Fadern kann lässt sich dann jeweils Position und Spanne für die Dimensionen Kanal, Tonhöhe und Zeit festlegen. Ist die Spanne für alle Dimensionen null, wird eine einzelne Aktivierung ausgewählt, sonst eine Gruppe. Mit einem zusätzlichen Taster kann festgelegt werden, ob die vorher ausgewählten Aktivierungen in der neuen Auswahl beibehalten oder ersetzt werden. So können schnell komplexe Aktivierungsmuster, wie in Abbildung 2 zu sehen, festgelegt werden, die zu interessanten klanglichen Ergebnissen führen. Andere steuerbare Parameter von Bedeutung sind unter Anderem die Lernrate der Optimierung, die Größe der zufälligen zeitlichen Verschiebungen des Inputs (große Verschiebungen führen zu flächigeren Klängen) und die Beimischung bereitgestellter Klänge.

Während der Performance hat man zwar ein gewisses Maß an Kontrolle, besonders interessant ist aber, dass es sich Interaktionen mit einem System handelt, das selbst ein (natürlich nur sehr eingeschränktes) Verständnis von Musik hat. In diesem Sinne kann man das System als Hybrid aus Instrument und Mitspieler betrachten.

Das hier beschriebene Setup ist sehr flexibel, problemlos können Netzwerke verwendet werden, die mit anderen Daten trainiert wurden oder eine andere Architektur besitzen.

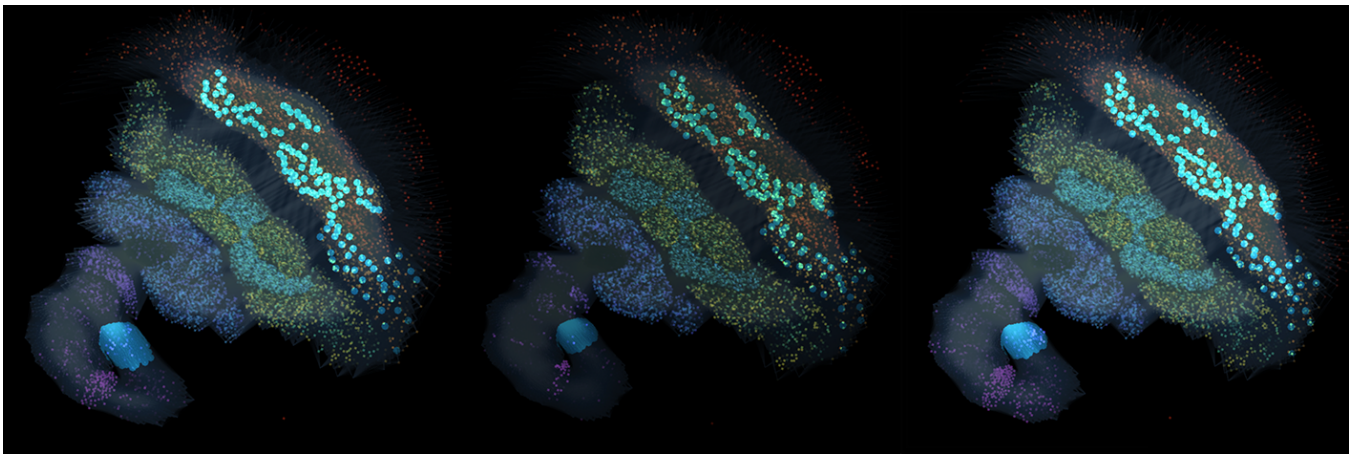

Abbildung 2: Live-Visualisierung der neuronalen Aktivierungen zu drei unterschiedlichen Zeitpunkten desselben Audioclips. Die hellen Bereiche sind fokussiert und werden durch das Optimierungsverfahren maximiert.

Auch während einer Performance können mehrere Netzwerke verwendet werden.

Das Generieren der Audioclips ist rechenintensiv. Für interessante, sich nicht zu langsam entwickelnde Ergebnisse sind mehrere GPUs nötig. Glücklicherweise können die Berechnungen auf einen externen Server ausgelagert werden, da Latenzen keine große Rolle spielen (es werden immer viersekündige Blocks abgespielt, die Kontrollparameter haben also ohnehin keine ganz sofortige Auswirkung). Somit ist eine normale Internetverbindung ausreichend und der Server kann bei Cloud-Computing-Anbietern angemietet werden.

Auf der Seite https://vincentherrmann.github.io/demos/immersions sind Audio- und Videodemos verfügbar.

LITERATUR

[1] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Brandon Tran, and Aleksander Madry. 2019. Learning Perceptually-Aligned Representations via Adversarial Robustness. *arXiv preprint arXiv:1906.00945* (2019).

[2] Karl Friston. 2005. A theory of cortical responses. *Philosophical transactions of the Royal Society B: Biological sciences* 360, 1456 (2005), 815–836.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 770–778.

[4] Yifan Hu. 2005. Efficient, high-quality force-directed graph drawing. *Mathematica Journal* 10, 1 (2005), 37–71.

[5] Alexander JE Kell, Daniel LK Yamins, Erica N Shook, Sam V Norman-Haignere, and Josh H McDermott. 2018. A task-optimized neural network replicates human auditory behavior, predicts brain responses, and reveals a cortical processing hierarchy. *Neuron* 98, 3 (2018), 630–644.

[6] Alexander Mordvintsev, Christopher Olah, and Mike Tyka. 2015. Inceptionism: Going deeper into neural networks. (2015).

[7] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. 2017. Feature visualization. *Distill* 2, 11 (2017), e7.

[8] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).